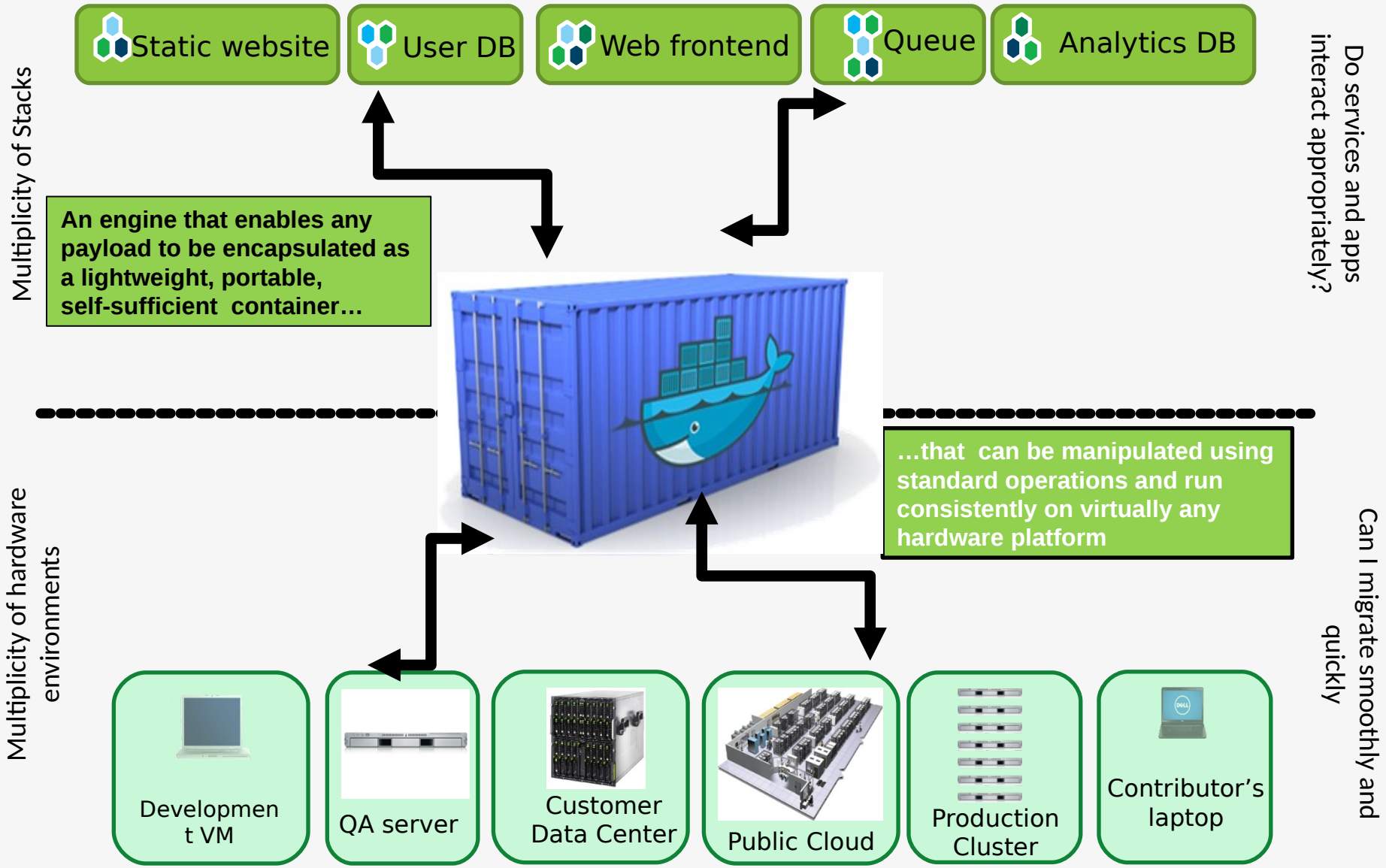
Java and Docker

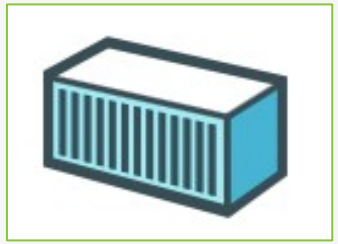
Dinakar Guniguntala, Architect, Java Cloud Optimization

Docker is a shipping container system for code



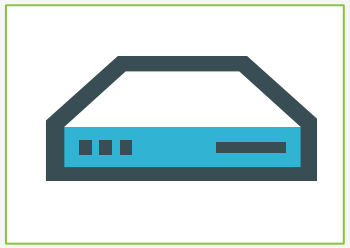
Docker Basics

Image
- A read-only snapshot of a container stored in Docker Hub to be used as a template for building containers



Container
- The standard unit in which the application service resides or transported

Docker Hub
- Available in SaaS or Enterprise to deploy anywhere you choose
- Stores, distributes and shares container images



Docker Engine
- A program that creates, ships and runs application containers
- Runs on any physical and virtual machine or server locally, in private or public cloud
- Client communicates with Engine to execute commands

Docker Basics Contd

Isolation

- Docker uses several Linux kernel features such as namespaces (ipc, mount, pid, network and user), Apparmor and SELinux profiles and control groups (cgroups), chroot on steroids to achieve Isolation

Lightweight

- Containers are “light” users of system resources, much smaller than VMs, startup time is less than a second and have better performance

User-friendly

- Developers build with ease and ship higher-quality applications
- Sysadmins deploy workload based on business priorities and policies

Dockerfile basics

- Layers – aufs (or other unionFS)
- $ docker history
- Docker image size
  - push times
  - provisioning times
  - deploy times
  - storage
  - security
- Alpine Linux, SFJ images

Java Container Startup Time

- Shared Class Cache
  - -Xshareclasses:name=appcache
  - java -Xshareclasses:name=liberty,cacheDir=/opt/ibm/wlp/output/.classCache,printallstats
- Liberty startup down to 3.x seconds from 4.x seconds
  - 20 – 30% reduction in startup time with SCC
- Run application in typical workload scenarios with SCC turned on.
  - Gather classes loaded (using “printallstats”)
  - Bake SCC into base Docker Image.

CPU and Memory

- cgroups
  - $ systemd-cgls
  - $ systemd-cgtop
  - cpu.shares
- cpusets
  - cpus
  - mems
    - /sys/fs/cgroup/memory/memory.limit_in_bytes
    - /sys/fs/cgroup/memory/docker/memory.soft_limit_in_bytes
  - java -verbose:sizes -version

- Docker options

```
--cpu-percent int          CPU percent (Windows only)
--cpu-period int           Limit CPU CFS period
--cpu-quota int            Limit CPU CFS quota
-c, --cpu-shares int       CPU shares (relative weight)
--cpuset-cpus string       CPUs in which to allow execution (0-3, 0,1)
--cpuset-mems string       MEMs in which to allow execution (0-3, 0,1)

--memory string                  Memory limit
--memory-reservation string      Memory soft limit
--memory-swap string             Swap limit equal to memory plus swap
                                 '-1' to enable unlimited swap
--memory-swappiness int          Tune container memory swappiness (0 to 100)
                                 (default -1)
```

The Idle problem

- 30% servers are sitting idle in the cloud*

- Charging model on most clouds based on memory usage (GB/hr on Bluemix).
- Historically poor Java idle behavior causes CPU burn.
  - Starves other JVM instances.
  - Increased costs for CPU usage.

- Java is currently inefficient with heap usage on Idle.
  - No reduction in memory usage even when on long periods of idle.
  - Big data workloads cause heap expansion but don't contract on Idle.

- Might hurt application performance on Active after a long period of Idle as heap cleanup may be needed.

- First step is to be able to measure precise JVM CPU and Memory usage.

* https://www.forbes.com/sites/benkepes/2015/06/03/30-of-servers-are-sitting-comatose-according-to-research/#39b31e8d59c7

JvmCpuMonitorInfo

JvmCpuMonitorInfo()
Creates a new JvmCpuMonitorInfo instance.

Method Summary

All Methods | Static Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
boolean	**equals**(java.lang.Object obj)
static JvmCpuMonitorInfo	**from**(javax.management.openmbean.CompositeData cd) Receives a CompositeData representing a **JvmCpuMonitorInfo** object and attempts to return the root **JvmCpuMonitorInfo** instance.
long	**getApplicationCpuTime**() This method returns the total CPU usage for all application threads.
long[]	**getApplicationUserCpuTime**() This method returns an array of CPU usage for all user defined thread categories.
long	**getGcCpuTime**() This method returns the total CPU usage of all GC threads.
long	**getJitCpuTime**() This method returns the total CPU usage of all JIT Threads.
long	**getResourceMonitorCpuTime**() This method returns the total CPU usage for all threads of the "Resource-Monitor" category.
long	**getSystemJvmCpuTime**() This method returns the total CPU usage of the "System-JVM" category, which includes GC, JIT and other JVM daemon threads.
long	**getTimestamp**() This method returns the last sampling time stamp.
int	**hashCode**()
java.lang.String	**toString**() Text description of this **JvmCpuMonitorInfo** object.

-Xtune:virtualized

- Reduces JVM CPU consumption when Idle.
- Needs a large shared class cache to maintain peak performance.
- AoT space in the Shared Class Cache (SCC) must not be capped.

Logging and Monitoring

- Logging
  - Docker Volume Container (DVC)
    - --volumes-from
    - docker create -v /logdata --name logdvc websphere-liberty /bin/true

- Monitoring
  - $ docker top
  - $ docker stats
  - $ docker inspect

QUESTIONS ?